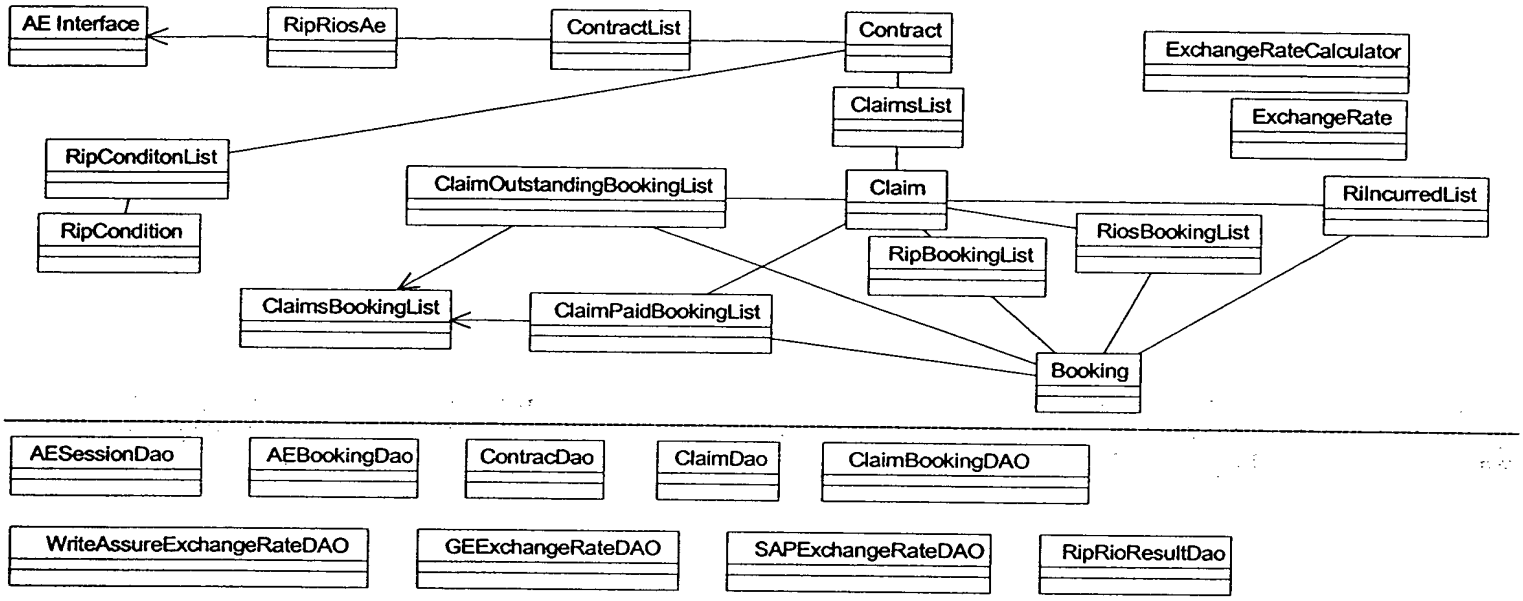
3.3 Business Objects

The Business model provides the core RIP/RIOS services.



3.4 BO Tables ,BUSINESS_ADDL_WA,RVREPE

Writasure exports a CSV file with 1 Mio bookings. We load this table in a stage table RVREPE and insert the amounts in USD in a 1:1 table. We also use a table Business_addl_wa which belongs to the RDB.

The Core entities A Contract, A Claim, Bookings and the Reinstatement conditions are extracted from the table Business_addl_wa and the RVREPE. The Only Claims Paid or Claims OS bookings (and Claims & Contracts) are considered. This selection reduces the effort for data access.

Each BO Table is access through a Data Access Object (DAO).

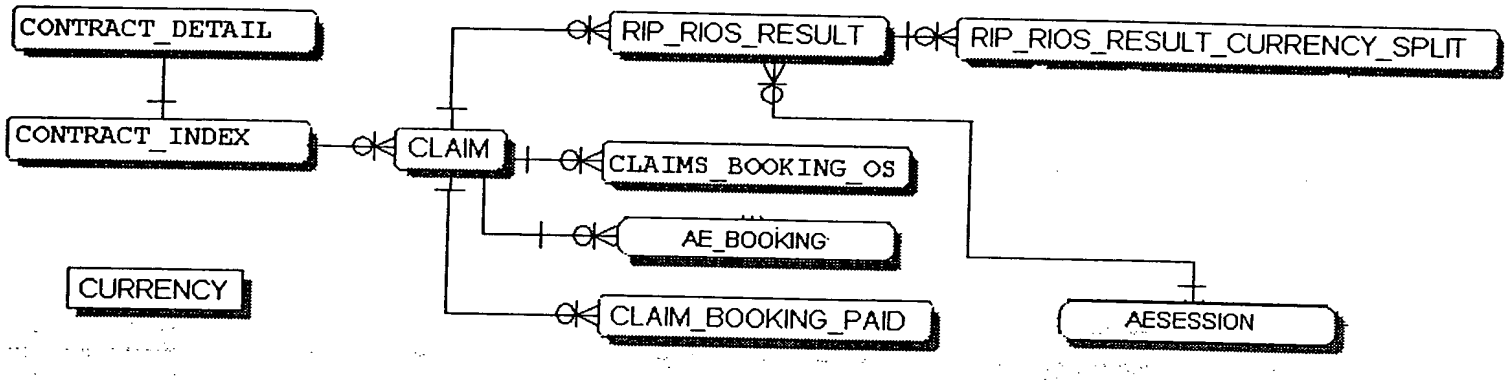
[redacted]

Appendix C

C-1

[redacted]

4 Database


CONTRACT_DETAIL
CONTRACT_INDEX
CLAIM
RIP_RIOS_RESULT
RIP_RIOS_RESULT_CURRENCY_SPLIT
CLAIMS_BOOKING_OS
AE_BOOKING
CURRENCY
CLAIM_BOOKING_PAID
AESESSION

4.1 SQL's

QUERY	Usage Count	Performance	Index Strategy	Purpose of the Query.
SELECT CI.MAJORCLASS_NAME MAJORCLASS, CI.CONTRACT_NBR CONTRACT, CI.UWYR_YR UWYR, CI.VERSION_NAME VERSION, CI.ENDORSE_NAME ENDORSE, CI.CONTREF_ID CONTREF, CTR_DTL.MAXLIMIT_AMT MAXLIMIT, CTR_DTL.LIMITOS_AMT LIMITOS, CTR_DTL.PREMIUMOS_AMT PREMIUMOS, CTR_DTL.DEDUCKTOS_AMT DEDUCKTOS, CTR_DTL.QUOTESHARE_AMT QUOTESHARE, CTR_DTL.CONTRACTCURRENCY_CD CONTRACTCURRENCY, CTR_DTL.BROKER_NAME BROKER, CTR_DTL.CEDENT_NAME CEDENT, CTR_DTL.INCEPTIONDATE_DATE INCEPTIONDATE, CTR_DTL.EXPIRYDATE_DATE EXPIRYDATE FROM CONTRACT_INDEX CI , CONTRACT_DETAIL CTR_DTL WHERE FK_LOAD_ID = <<loadId>> AND CI.MAJORCLASS_NAME = CTR_DTL.MAJORCLASS_NAME AND CI.CONTRACT_NBR = CTR_DTL.CONTRACT_NBR AND CI.UWYR_YR = CTR_DTL.UWYR_YR AND CI.VERSION_NAME = CTR_DTL.VERSION_NAME AND CI.ENDORSE_NAME = CTR_DTL.ENDORSE_NAME Incase of filters - AND CI.MAJORCLASS_NAME = <<majorclass>> AND CI.CONTRACT_NBR = <<contractNbr>>AND CI.UWYR_YR = <<UwYear>> AND CI.VERSION_NAME =<<versionName>> AND CI.ENDORSE_NAME = <<endorseName>> AND CI.CONTREF_ID = <<contrefId>> would be added to the WHERE clause depending upon the filters passed.	1	HI	The indexes on the primary keys of the contract_detail and the contract_indare the critical indices for this fetch. The indices involved are CreateIContractContRefIndex.sql, CreateIContractDetailKey.sql, CreateIContractPrimKeyIndex.sql, CreateICtrtDtlContractPrimaryKey.sql,	This query is to fetch all the contracts from the CONTRACT_DETAIL tables that are supposed to be present in the CONTRACT_INDEX table for a given load Id. Incase of a filter wherein the user is given the facility of passing any or all of the primary keys the fields would be accordingly added to the WHERE clause.

SELECT C.PERIL,C.DATEOFLOSS, NVL(C.CATCODE, 'none') CATCODE ,C.CLAIMNO FROM **CLAIM** C WHERE C.FK_MAJORCLASS = <<majorClass>> AND C.FK_CONTRACT = <<contractNbr>> AND C.FK_UWYR = <<uwyr>> AND C.FK_VERSION = <<version>> AND C.FK_ENDORSE = <<endorse>> ORDER BY C.CLAIMNO, C.CATCODE, C.DATEOFLOSS	>5282	MED	The indices involved are CreateIClaimCatcodeIndex.sql, CreateIClaimContractPrimKeyIndex.sql, CreateIClaimPrimaryKey.sql	This query fetches the claim from the CLAIM table for a specific contract.
SELECT B.ID,B.ORIGCCY, B.BILLMON, B.BILLYEAR ,B.AMOUNT FROM **CLAIMS_BOOKING_PAID B or CLAIMS_BOOKING_OS B** WHERE ((B.BILLMON <= <<billMon>> AND B.BILLYEAR = <<billYear>>) OR B.BILLYEAR < <<billYear>>) AND B.FK_CLAIMNO = <<claimNo>> AND B.FK_DATEOFLOSS = <<dateOfLoss>> AND B.FK_CATCODE = <<catCode>> AND B.FK_MAJORCLASS = <<majorClass>> AND B.FK_CONTRACT = <<contractNbr>> AND B.FK_UWYR = <<uwyr>> AND B.FK_VERSION = <<version>> AND B.FK_ENDORSE = <<endorse>> ORDER BY B.FK_DATEOFLOSS, B.FK_CATCODE, B.ORIGCCY	Approx **5282 (contracts) * 24419 (claims)**	MED	The indices involved are CreateIClaimsBookOSCatcodeIndex.sql, CreateIClaimsBookOSContractPrimKeyIndex.sql, CreateIClaimsBookOSKenzIndex.sql, CreateIClaimsBookPaidCatcodeIndex.sql, CreateIClaimsBookPaidContractPrimKeyIndex.sql, CreateIClaimsBookPaidKenzIndex.sql	This query fetches the Paid and OS bookings for individual claims
INSERT INTO **RIP_RIOS_RESULT** (ID, TOTALRIP, TOTALRIOS, NETRIOS, FK_MAJORCLASS, FK_CONTRACT, FK_UWYR, FK_VERSION, FK_ENDORSE, FK_CONTREF, FK_CLAIMNO, FK_CATCODE, FK_DATEOFLOSS, FK_PERIL, FK_SESSION) VALUES (<<id>>, <<totalrip>>, <<totalrios>>, <<netrios>>, <<fk_mmajorclass>>, <<fk_contract>>, <<fk_uwyr>>, <<fk_version>>, <<fk_endorse>>, <<fk_contref>>, <<fk_claimnno>>, <<fk_catcode>>, <<fk_dateofloss>>, <<fk_peril>>, <<fk_session>>)	> **24419 claims**	HI		This query is supposed to insert into the rip_rios_result table after the calculations are over, to document the results.

[redacted] [redacted]

INSERT INTO **RIP_RIOS_RESULT_CURRENCY_SPLIT** (ID, RITYPE, AMOUNT, ISOCODE, RISPLIT, FK_RIP_RIOS_RESULT, FK_SESSION) VALUES (RIP_RIOS_RESULT_CURR_SPLIT_SEQ.nextval, <<id>>, <<ritype>>, <<amount>>, <<isocode>>, <<ripsplit >>, <<fk_rip_rios_result>>, <<fk_session>>)	> 48820	**HI**		This query inserts into the rip_rios_currency_split after the calculations are over.
SELECT REINST_NBR, REINST_PCT FROM **REINSTATEMENT_CONDITIONS** WHERE FK_LOAD_ID = <<loadid>> AND FK_CONTREF_ID = <<contref>> AND (REINST_NBR <> 0 OR REINST_PCT <> 0) ORDER BY REINST_ID	**>5282 contr acts**	**HI(less than 500 ms)**	The indices involved are CreateIRICondFkContrefKey.sql	This query fetches the Rinumber and the Ripercentage for all the RIConditions for a particular contract, in a particular load.
SELECT **COUNT(*)** FROM **REINSTATEMENT_CONDITIONS** WHERE FK_LOAD_ID = <<loadid>> AND FK_CONTREF_ID = <<contref>> AND (REINST_NBR <> 0 OR REINST_PCT <> 0) ORDER BY REINST_ID	**>5282 contr acts**	**HI(less than 150ms)**	The indices involved are CreateIRICondFkContrefKey.sql	This query finds out the number of Riconditions for a particular contract in a particular load.

[redacted] [redacted]

INSERT INTO **LTC_REPORT.AE_BOOKING** (ID, FK_SESSION, SOURCE, REINSURE, BOOK_ID, UW_YEAR, OCC_YEAR, ACCOUNT_YEAR, ACCOUNT_PERIOD, BYRP, BOOK_BRANCH, BOOKCODE, ORIG_CURR, AMOUNT, SIGN, A_TYPE, CLAIM_ID, POOL_ID, REFNO, REF_TYPE, COMPUTER_DATE, POSTING_DATE, TREATYNO, IN_OUT, DIR_INDIR, BRANCH, POOL_RE, CEDENT, BROKER, SAP_COMP, BUS_AREA, AGG_CODE, SAP_TYPE, SAP_BRANCH, TRADING_PART, BANK_ACCOUNT, SAP_CUR, SAP_AGGR, ORIG_COMPUTER_DATE, LIRMA_REF, LIRMA_F) VALUES (<<id>>, <<fk_session>>, <<source>>, <<reinsure>>, <<book_id>>, <<uw_year>>, <<occ_year>>, <<account_year>>, <<account_period>>, <<byrp, book_branch>>, <<bookcode>>, <<orig_curr>>, <<amount>>, <<sign>>, <<a_type>>, <<claim_id>>, <<pool_id>>, <<refno>>, <<ref_type>>, <<computer_date>>, <<posting_date>>, <<treatyno>>, <<in_out>>, <<dir_indir>>, <<branch>>, <<pool_re>>, <<cedent>>, <<broker>>, <<sap_comp>>, <<bus_area>>, <<agg_code>>, <<sap_type>>, <<sap_branch>>, <<trading_part>>, <<bank_account>>, <<sap_cur>>, <<sap_aggr>>, <<orig_computer_date>>, <<lirma_ref>>, <<lirma_f>>)	**>4882 0 without exceptions**	**HI**		This query inserts bookings into the AE_Booking table.

[redacted] [redacted]

INSERT INTO **AESESSION** (ID, DATEOFRUN, TIMETAG, USERNAME, SOFTWAREVERSION, SESSIONTYPE, USERCOMMENT, FK_LOAD_ID) VALUES (<<id>>, <<dateofrun>>, <<timetag>>, <<username>>, <<softwareversion>>, <<sessiontype>>, <<usercomment>>, <<fk_load_id>>)	1	HI		This query inserts the session details into the AESession table.
SELECT ID, TOTALRIP, TOTALRIOS, NETRIOS, FK_MAJORCLASS, FK_CONTRACT, FK_UWYR, FK_VERSION, FK_ENDORSE, FK_CONTREF, FK_CLAIMNO, FK_CATCODE, FK_DATEOFLOSS, FK_PERIL,FK_SESSION FROM **RIP_RIOS_RESULT** WHERE FK_SESSION = <<session>>	1	HI		This query is used to select from rip_rios_result table
SELECT EXCH_RATE_YR_D, PRD_NUM_Q, CUR_C,EXCH_RATE_PRD_C,EXCH_Z FROM **GE_EXCHANGE_RATE** WHERE ORI_OF_EXCH_RATE_C =' GE' AND EXCH_RATE_TYP_C = ' A' AND TO_NUMBER(EXCH_RATE_YR_D||LPAD(PRD_NUM_Q,5,0)) = (SELECT MIN(TO_NUMBER(EXCH_RATE_YR_D||LPAD(PRD_NUM_Q,5,0))) FROM **GE_EXCHANGE_RATE** WHERE EXCH_RATE_PRD_C = << exch_rate_prd_c>> AND CUR_C = <<cur_c>> AND BSIS_CUR_C ='USD' AND ORI_OF_EXCH_RATE_C =' GE' AND EXCH_RATE_TYP_C = 'A')		MI		This query fetches the oldest entry in the GE_EXCHANGE_RATE table specific to the isocode and currency type, specified.

SELECT EXCH_RATE_YR_D, PRD_NUM_Q, CUR_C,EXCH_RATE_PRD_C,EXCH_Z FROM **GE_EXCHANGE_RATE** WHERE ORI_OF_EXCH_RATE_C =' GE' AND EXCH_RATE_TYP_C = 'A' ORDER BY EXCH_RATE_YR_D, PRD_NUM_Q, CUR_C, EXCH_RATE_PRD_C,EXCH_Z		**HI**		This query fetches the data from ge_exchange_rate table.
SELECT YEAR, QUARTER, ISOCODE, CURCODE,RATE FROM **SAP_RATES** ORDER BY YEAR, QUARTER, ISOCODE, CURCODE, RATE		**MI**		This query fetches the data from sap_rate table.
SELECT YEAR,QUARTER,ISOCODE,CURCODE FROM **SAP_RATES** WHERE CURCODE = <<curcode>> AND ISOCODE = <<isocode>> AND (YEAR||QUARTER) = (SELECT (MIN (YEAR||QUARTER)) FROM **SAP_RATES** WHERE CURCODE = <<curcode>> AND ISOCODE = <<isocode>>)		**HI**		This query selects the oldest entry from the sap_rates table for the specific isocode and currency
SELECT YEAR, MONTH, ISOCODE, CURRENCYTYPE, RATE FROM **CURRENCY** ORDER BY YEAR, MONTH, ISOCODE, CURRENCYTYPE,RATE		**LO**		This query fetches data from the currency table.

SELECT YEAR,MONTH,ISOCODE,CURRENCYTYPE FROM CURRENCY WHERE CURRENCYTYPE = <<currency_type>> AND ISOCODE = <<isocode>> AND TO_NUMBER(LPAD(YEAR,4,0)||LPAD(MONTH,2,0)) = (SELECT MIN(TO_NUMBER(LPAD(YEAR,4,0)||LPAD(MONTH,2,0))) FROM CURRENCY WHERE CURRENCYTYPE = <<currency_type>> AND ISOCODE = <<isocode>>)		ME		This query fetches the oldest record from the CURRENCY table from the given isocode and currency type.
INSERT INTO LTC_REPORT.USER_INPUT (ID, SESSIONID, INPUTPARAMETERNAME, INPUTPARAMETERVALUE) VALUES (<<id>>, <<sessionid>>, <<inputparametername>>, <<inputparametervalue>>)	1	HI		This query inserts the user specified parameters for the calculator run into the User_Input table
SELECT NVL(MAX(ID),-1) SessionId FROM AESESSION WHERE SESSION_STATUS_IND = 'SUCCESS' AND SESSIONTYPE = <<sessionType>> AND CALCULATOR_TYPE = <<Calculatortype>>	1	ME		This query gets the previous session id for the purpose of movement calculation.
UPDATE AESESSION SET SESSION_STATUS_IND = <<:status>> WHERE ID = <<id>>	3			This query is used to update the session status to the aesession table
SELECT ID FROM AESESSION WHERE ID = <<id>> AND SESSION_STATUS_IND = 'SUCCESS' AND CALCULATOR_TYPE = <<calculatortype>>	1	HI		This query validates the user specified session id.

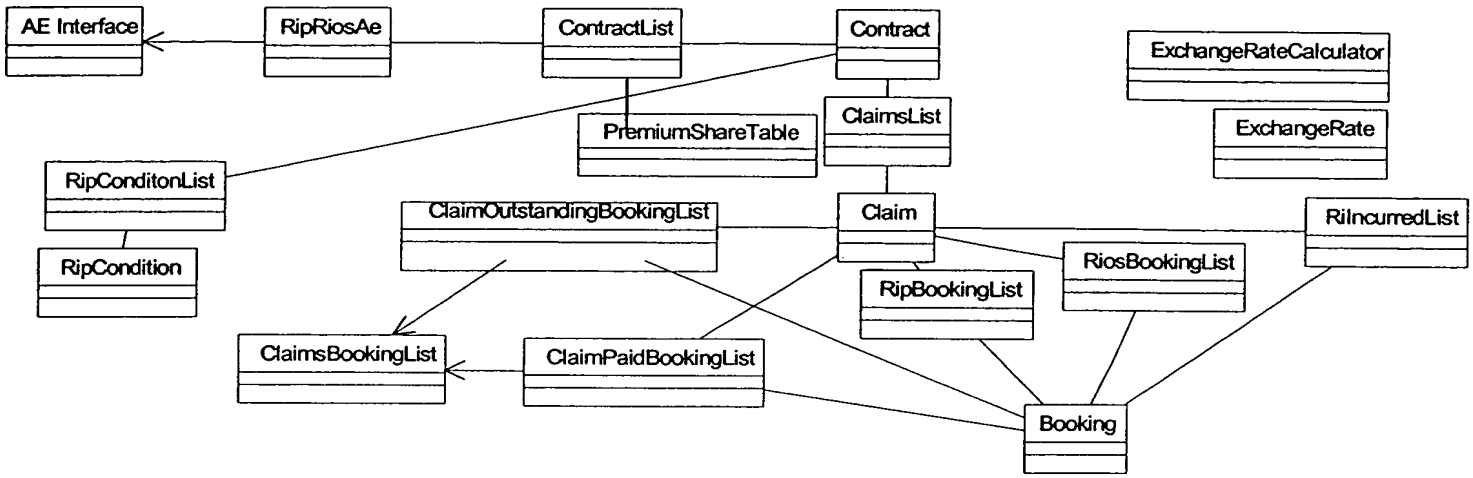
SELECT AMOUNT, RITYPE, ISOCODE FROM RIP_RIOS_RESULT A, RIP_RIOS_RESULT_CURRENCY_SPLIT B WHERE A.ID = B.FK_RIP_RIOS_RESULT AND A.FK_SESSION = <<fkSession>> AND A.FK_CONTREF = <<contref>> AND A.FK_CATCODE = <<catcode>> AND A.FK_CLAIMNO = <<claimno>> AND A.FK_DATEOFLOSS = <<dol>> AND B.ISOCODE = <<isocode>> AND B.RITYPE = <<ritype>>	For every booking once	Hi		This query is used to fetch the previous data for the purpose of movement calculations.
CREATE INDEX i_CLAIM_catcode ON CLAIM (catcode)				This query creates an index on the catcode field of CLAIM
CREATE INDEX i_CLAIM_contractprimkey ON CLAIM (FK_majorclass, FK_contract, FK_uwyr, FK_version, FK_endorse)				This query creates an index on the primary keys of the claim.
CREATE INDEX i_CLAIM_CLAIMPRIMKEY ON CLAIM (claimno, catcode, dateofloss)				This query creates an index on the primary keys of the claim
CREATE INDEX i_CONTRACT_contref ON CONTRACT_INDEX (contref_id)				Creates an index on the Conref field of the CONTRACT_INDEX table
CREATE INDEX I_ContractDetailKey ON LTC_REPORT.CONTRACT_DETAIL(CONTRACT_DETAIL_ID)				Creates and index on the contract_detail_id of the CONTRACT_DETAIL table.
CREATE INDEX i_contract_primkey ON CONTRACT_INDEX (majorclass_name, contract_nbr, uwyr_yr, version_name, endorse_name)				Creates index on the primary key of the CONTRACT_INDEX table
CREATE INDEX i_CLAIMS_BOOKING_OS_catcode ON CLAIMS_BOOKING_OS (FK_catcode)				Creates an index on the catcode field of the CLAIMS_BOOKING_OS
CREATE INDEX i_CB_OS_contractprimkey ON CLAIMS_BOOKING_OS (FK_majorclass, FK_contract, FK_uwyr, FK_version, FK_endorse)				Creates an index on the contract primary keys of the CLAIMS_BOOKING_OS
CREATE INDEX i_CLAIMS_BOOKING_OS_kenz ON CLAIMS_BOOKING_OS (kenz)				Creates an index on the kenz field of the CLAIMS_BOOKING_OS

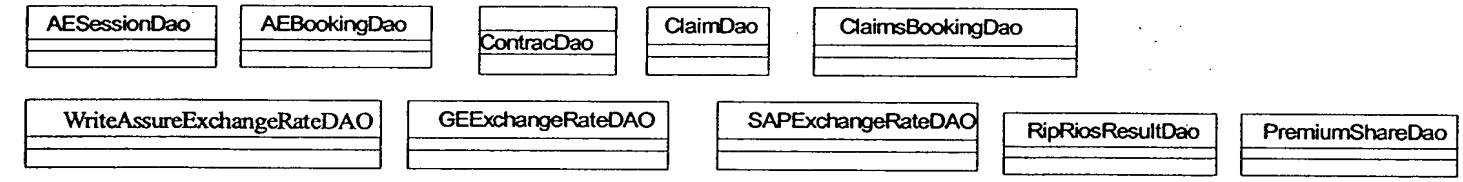
[redacted] [redacted]

CREATE INDEX i_CLAIMS_BOOKING_PAID_catcode ON CLAIMS_BOOKING_PAID (FK_catcode)				Creates an index on the catcode field of the CLAIMS_BOOKING_PAID
CREATE INDEX i_CB_PAID_contractprimkey ON CLAIMS_BOOKING_PAID (FK_majorclass, FK_contract, FK_uwyr, FK_version, FK_endorse)				Creates an index on the contract primary keys of the CLAIMS_BOOKING_PAID
CREATE INDEX i_CLAIMS_BOOKING_PAID_kenz ON CLAIMS_BOOKING_OS (kenz)				Creates an index on the kenz field of the CLAIMS_BOOKING_PAID
CREATE INDEX I_RICondFkContrefKey ON REINSTATEMENT_CONDITIONS (FK_CONTREF_ID);				Creates an index on the fk_contref_id field of the REINSTATEMENT_CONDITIONS table

[redacted] [redacted]

5 Objects


AE Interface
RipRiosAe
ContractList
Contract
ExchangeRateCalculator
PremiumShareTable
ClaimsList
ExchangeRate
RipConditonList
ClaimOutstandingBookingList
Claim
RiIncurredList
RipCondition
RiosBookingList
RipBookingList
ClaimsBookingList
ClaimPaidBookingList
Booking


AESessionDao
AEBookingDao
ContracDao
ClaimDao
ClaimsBookingDao
WriteAssureExchangeRateDAO
GEExchangeRateDAO
SAPExchangeRateDAO
RipRiosResultDao
PremiumShareDao

6 RIPRIOS Calculator Method Specification

This section gives the specifications for the different classes involved in the RipRios Calculator. The classes have been divided into the following major categories:

- **Method Objects.**

  This class will drive the business logic of the RipRios Calculator. It can be considered as the engine for the calculator. The following control class is identified for the calculator.

  1. RipRiosAEController
  2. RipRiosAE

- **Business Objects.**

These classes will form the business objects for the calculator. Each class provides an independent functionality. The functionality provided is invoked by them method objects or by some other business objects. The following are the main classes and Interfaces identified.

1. Contract
2. ContractList
3. InputContractFilter
4. Claim
5. ClaimList
6. ClaimOutStandingBookingList
7. ClaimPaidBookingList
8. Booking
9. InputBooking
10. EstimatedBooking
11. ClaimPaidBookingList
12. ClaimBookingList
13. RIIncurredBooking
14. RIIncurredBookingList
15. RIOSBooking
16. RIOSBookingList
17. RIPBooking
18. RIPBookingList
19. RIPCondition
20. RIPConditionList

21. BusinessObject
22. ExchangeRateCalculator
23. ExchangeRate
24. WriteAssureExchangeRate
25. GEExchangeRate
26. SAPExchangeRate
27. AESession
28. AEBooking
29. AEBookingList
30. AggregatedBooking
31. Auditable
32. MovementController
33. Result (Interface)

- **DataAccess Objects**

1. ContractDAO
2. RIConditionDAO
3. ClaimDAO
4. ClaimBookingDAO
5. PremiumShareDAO
6. JDBCConnectionConfiguration
7. JDBCConnectionConfigurationFactory
8. AbstractDAO
9. AESessionDAO
10. AEBookingDAO
11. RipRiosResultDAO
12. ExchangeRateBase
13. ExchangeRateDAO
14. GEExchangeRateDAO
15. SAPExchangeRateDAO
16. WriteAssureExchangeRateDAO

[redacted] [redacted]

- **Policy Objects**
  1. GenericPolicy
  2. ExceptionPolicy
  3. HistoryPolicy
  4. IncrementalPolicy
  5. SapRatePolicy

6.1 RipRios Life Cycle Mode

This mode deals with the incremental calculation of the Rip and Rios values for a contract over its entire life, month by month.

This mode makes use of a Controller (the RipRiosAEController) to invoke the RipRiosAE over the range of cut off dates.

The start date and the end date provided by the use are used to calculate the list of cut-off dates. These cut-off dates are fed to the Accounting Engine, one by one.

The controller generates RipRiosAEController.xml. This XML contains the details, such as start date, end date, session id and previous session id, about each AE run.

7 Tests

7.1 Test Cases

Specification	**TestCaseWithVariance.xls.**		
JUnit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests ContractsWithIncorrectRIUsed		
Tier	Server		
Developer	Vinod[redacted]		
Total Number	1		
Log	Vinod [redacted]	[redacted]	1 OK
To Do			

Specification	**TestCaseWithVariance.xls.**		
JUnit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests ContractsWithVarianceM319_93		
Tier	Server		
Developer	Vinod[redacted]		
Total Number	1		
Log	Vinod [redacted]	[redacted]	1 OK
To Do			

[redacted]

Specification	**TestCaseWithVariance.xls.**		
JUnit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests ContractsWithVarianceN449		
Tier	Server		
Developer	Vinod [redacted]		
Total Number	1		
Log	Vinod [redacted]		1 OK
To Do			

Specification	**Patni**		
JUnit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests ContractsWithVarianceN5705_93_0_00		
Tier	Server		
Developer	Vinod [redacted]		
Total Number	1		
Log	Vinod [redacted]		1 OK
To Do			

[redacted]

[redacted]

Specification	**Patni**			
JUnit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests ContractsWithVarianceN6595_97_0_00			
Tier	Server			
Developer	Vinod Khader			
Total Number	1			
Log	Vinod [redacted]	[redacted]		1 OK
To Do				

Specification	**TestCaseWithVariance.xls.**			
JUnit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests ContractsWithVarianceN7427			
Tier	Server			
Developer	Vinod [redacted]			
Total Number	1			
Log	Vinod [redacted]	[redacted]		1 OK
To Do				

[redacted]

Specification	**TestCaseWithVariance.xls.**			
JUnit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests ContractsWithVarianceNew5Sep			
Tier	Server			
Developer	Vinod [redacted]			
Total Number	16			
Log	Vinod Khader	[redacted]		16 OK
To Do				

Specification	**New_Jana_Testcases.xls.**			
JUnit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests **NewJanaTestCaseC**			
Tier	Server			
Developer	Vinod [redacted]			
Total Number	5			
Log	Vinod Khader	[redacted]		5 OK
To Do				

[redacted]

[redacted]

Specification	**New_Jana_Testcases.xls.**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests NewJanaTestCaseM			
Tier	Server			
Developer	Vinod Khader			
Total Number	5			
Log	Vinod [redacted]	[redacted]		5 OK
To Do				

Specification	**New_Jana_Testcases.xls.**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests NewJanaTestCaseN			
Tier	Server			
Developer	Vinod [redacted]			
Total Number	5			
Log	Vinod [redacted]	[redacted]		5 OK
To Do				

Specification	**New_Jana_Testcases.xls.**

[redacted]

Junit Test Class: Call Arguments:	Com.ge.gefre.ltca.ae.riprios.tests NewJanaTestCaseS			
Tier:	Server			
Developer:	Vinod [redacted]			
Total Number:	5			
Log:	Vinod [redacted]	[redacted]		5 OK
To Do:				

Specification:	**runResults20020821.xls.**			
Junit Test Class: Call Arguments:	Com.ge.gefre.ltca.ae.riprios.tests RipRiosC0000671998000			
Tier:	Server			
Developer:	Vinod [redacted]			
Total Number:	1			
Log:	Vinod [redacted]	[redacted]		1 OK
To Do:				

Specification:	**runResults20020821.xls.**
Junit Test Class:	Com.ge.gefre.ltca.ae.riprios.tests RipRiosC0013261999000

[redacted]

Call Arguments				
Tier	Server			
Developer	Vinod [redacted]			
Total Number	1			
Log	Vinod [redacted]	[redacted]		1 OK
To Do				

Specification	**Patni**			
Junit Test Class	Com.ge.gefre.ltca.ae.riprios.tests RipRiosCurrencySplit			
Call Arguments				
Tier	Server			
Developer	Vinod [redacted]			
Total Number	1			
Log	Vinod [redacted]	[redacted]		1 OK
To Do				

Specification	**Patni**
Junit Test Class	Com.ge.gefre.ltca.ae.riprios.tests RipRiosGefreTest
Call Arguments	
Tier	Server

[redacted]

Developer	Vinod [redacted]			
Total Number	1			
Log	Vinod [redacted]	[redacted]		1 OK
To Do				

Specification	**MoreRIOS_Eg.xls.**			
Junit Test Class Cal Arguments	Com.ge.gefre.ltca.ae.riprios.tests RipRiosJana			
Tier	Server			
Developer	Vinod [redacted]			
Total Number	9			
Log	Vinod [redacted]	[redacted]		9 OK
To Do				

Specification	**Patni**
Junit Test Class Cal Arguments	Com.ge.gefre.ltca.ae.riprios.tests RipRiosJanaExchRate
Tier	Server

[redacted]

Developer	Vinod [redacted]			
Total Number	9			
Log	Vinod [redacted]	[redacted]		9 OK
To Do				

Specification	**TestCaseDolAggregation.xls.**
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests RipRiosN0007091989000
Tier	Server
Developer	Vinod [redacted]
Total Number	1
Log	Vinod [redacted] 1 OK
To Do	

Specification	**TestCaseDolAggregation.xls.**
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests RipRiosN0007091989000Rip
Tier	Server

[redacted]

Developer	Vinod [redacted]			
Total Number	1			
Log	Vinod [redacted]	[redacted]		1 OK
To Do				

Specification	**runResults20020821.xls**
Junit Test Class / Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests RipRiosN0011961999001
Tier	Server
Developer	Vinod [redacted]
Total Number	1
Log	Vinod [redacted] [redacted] 1 OK
To Do	

Specification	**runResults20020821.xls.**
Junit Test Class / Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests RipRiosN002341992000
Tier	Server
Developer	Vinod [redacted]
Total Number	1

[redacted]

[redacted]

Log	Vinod [redacted]	[redacted]		1 OK
To Do				

Specification	**runResults20020821.xls.**
Junit Test Class	Com.ge.gefre.ltca.ae.riprios.tests
Call Arguments	RipRiosN002341992000Fake
Tier	Server
Developer	Vinod [redacted]
Total Number	1

Log	Vinod [redacted]	[redacted]		1 OK
To Do				

Specification	**runResults20020821.xls.**
Junit Test Class	Com.ge.gefre.ltca.ae.riprios.tests
Call Arguments	RipRiosN0051121999000
Tier	Server
Developer	Vinod [redacted]
Total Number	1

Log	Vinod [redacted]	[redacted]		1 OK

[redacted]

To Do	

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests RipRiosNegativeCurrencySplit			
Tier	Server			
Developer	Vinod [redacted]			
Total Number	1			
Log	Vinod [redacted]	[redacted]		1 OK
To Do				

Specification	**MoreRIOS_Eg_Patni.xls**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests RipRiosPatni			
Tier	Server			
Developer	Vinod [redacted]			
Total Number	17			
Log	Vinod [redacted]	[redacted]		17 OK
To Do				

[redacted]

Specification	**TestCaseDolAggregation.xls**		
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests TestBookingListAggregation		
Tier	Server		
Developer	Vinod [redacted]		
Total Number	1		
Log	Vinod [redacted]	[redacted]	1 OK
To Do			

Specification	**Patni**		
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests TestExceptionXml		
Tier	Server		
Developer	Vinod [redacted]		
Total Number	4		
Log	Vinod [redacted]	[redacted]	4 OK
To Do			

[redacted]

Specification	**Patni**		
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests TraceLogTest1		
Tier	Server		
Developer	Vinod [redacted]		
Total Number	6		
Log	Vinod [redacted]		6 OK
To Do			

Specification	**Patni**		
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests RipRiosC_1326_2001_0_00		
Tier	Server		
Developer	Vinod [redacted]		
Total Number	1		
Log	Vinod [redacted]		1 OK
To Do			

[redacted]

[redacted]

Specification	**Patni**		
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests RipRiosC_1385_1999_0_01		
Tier	Server		
Developer	Vinod [redacted]		
Total Number	1		
Log	Vinod [redacted]	[redacted]	1 OK
To Do			

Specification	**Patni**		
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests RipRiosM_679_2001_0_00		
Tier	Server		
Developer	Vinod [redacted]		
Total Number	1		
Log	Vinod [redacted]	[redacted]	1 OK
To Do			

[redacted]

[redacted]

Specification	**Testcases Sent by Florence**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests FlorenceTest			
Tier	Server			
Developer	Vinod [redacted]			
Total Number	12			
Log	Vinod [redacted]	[redacted]		12 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests TestNetRiosFunc			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	12			
Log	Lilly [redacted]	[redacted]		11 OK
To Do				

[redacted]

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.riprios.tests TestContractFetchWhereClause			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	12			
Log	Lilly [redacted]	[redacted]		11 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	com.ge.gefre.ltca.dao.tests AESessionDao			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	7			
Log	Lilly [redacted]	[redacted]		7 OK
To Do				

Specification	**Patni**
Junit Test Class Call Arguments	com.ge.gefre.ltca.dao.tests AllExchangeRateTablesDao

[redacted]

[redacted]

Tier	Server			
Developer	Lilly [redacted]			
Total Number	4			
Log	Lilly [redacted]	[redacted]		4 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests ClaimBookingOutStandingDao			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	4			
Log	Lilly [redacted]	[redacted]		4 OK
To Do				

Specification	**Patni**
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests ClaimBookingPaidDao
Tier	Server
Developer	Lilly [redacted]

[redacted]

[redacted]

Total Number	3			
Log	Lilly [redacted]	[redacted]		3 OK
To Do				

Specification	**Patni**
Junit Test Class	Com.ge.gefre.ltca.dao.tests ClaimDao
Call Arguments	
Tier	Server
Developer	Lilly [redacted]
Total Number	4
Log	Lilly [redacted] [redacted] 4 OK
To Do	

Specification	**Patni**
Junit Test Class	Com.ge.gefre.ltca.dao.tests ContractDao
Call Arguments	
Tier	Server
Developer	Lilly [redacted]
Total Number	6

Log	Lilly [redacted]	[redacted]		6 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests TestPremiumShareDao			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	6			
Log	Lilly [redacted]	[redacted]		6 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests GEExchangeHistoryPolicyDaoTest			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	20			
Log	Lilly [redacted]	[redacted]		20 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests LtcTestCase			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	0			
Log	Lilly [redacted]	[redacted]		0 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests RipRiosBugFixTest			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	1			
Log	Lilly [redacted]	[redacted]		1 OK
To Do				

Specification	**Patni**

[redacted]

[redacted]

Junit Test Class / Call Arguments	Com.ge.gefre.ltca.dao.tests SAPExchangeHistoryPolicyDaoTest			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	10			
Log	Lilly [redacted]	[redacted]		10 OK
To Do				

Specification	**Patni**			
Junit Test Class / Call Arguments	Com.ge.gefre.ltca.dao.tests TestAEBookingDAO			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	9			
Log	Lilly [redacted]	[redacted]		9 OK
To Do				

Junit Test Class / Description	Com.ge.gefre.ltca.dao.tests TestGEDefPolicy This testprogram is created to test the Default Policy for GE exchange rates table.

[redacted]

Call Arguments				
Tier	Server			
Developer	Lilly [redacted]			
Total Number	9			
Log	Lilly [redacted]	[redacted]		9 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests TestGEExcepPolicy			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	13			
Log	Lilly [redacted]	[redacted]		13 OK
To Do				

Specification	**Patni**
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests TestGEHistory
Tier	Server

[redacted]

[redacted]

Developer	Lilly [redacted]			
Total Number	36			
Log	Lilly [redacted]	[redacted]		36 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests TestGEIncrPolicy			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	10			
Log	Lilly [redacted]	[redacted]		10 OK
To Do				

Specification	**Patni**
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests TestMonthChange
Tier	Server
Developer	Lilly [redacted]

[redacted]

[redacted]

Total Number	2			
Log	Lilly [redacted]	[redacted]		2 OK
To Do				

Specification	**Patni**
Junit Test Class	Com.ge.gefre.ltca.dao.tests TestSAPDefPolicy
Description	This testprogram is created to test the Default Policy for SAP_Rates exchange rates table.
Call Arguments	
Tier	Server
Developer	Lilly [redacted]
Total Number	7

Log	Lilly [redacted]	[redacted]		7 OK
To Do				

Specification	**Patni**
Junit Test Class	Com.ge.gefre.ltca.dao.tests TestSAPExcepPolicy
Call Arguments	
Tier	Server
Developer	Lilly [redacted]
Total Number	7

[redacted]

[redacted]

Log	Lilly [redacted]	[redacted]		7 OK
To Do				

Specification	**Patni**			
Junit Test Class	Com.ge.gefre.ltca.dao.tests TestSAPHistory			
Cal Arguments				
Tier	Server			
Developer	Lilly [redacted]			
Total Number	22			
Log	Lilly [redacted]	04-Sep-02		22 OK
To Do				

Specification	**Patni**			
Junit Test Class	Com.ge.gefre.ltca.dao.tests TestSapRatesIncrementalPolicy			
Cal Arguments				
Tier	Server			
Developer	Lilly [redacted]			
Total Number	15			
Log	Lilly [redacted]	[redacted]		15 OK

[redacted]

[redacted]

To Do	

Specification	**Patni**			
Junit Test Class	Com.ge.gefre.ltca.dao.tests WriteAssureDefaultTest			
Call Arguments				
Tier	Server			
Developer	Lilly [redacted]			
Total Number	8			
Log	Lilly [redacted]	[redacted]		8 OK
To Do				

Specification	**Patni**			
Junit Test Class	Com.ge.gefre.ltca.dao.tests WriteAssureExceptionTest			
Call Arguments				
Tier	Server			
Developer	Lilly [redacted]			
Total Number	8			
Log	Lilly [redacted]	[redacted]		8 OK
To Do				

Specification	**Patni**
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests WriteAssureHistoryTest
Tier	Server
Developer	Lilly [redacted]
Total Number	28
Log	Lilly [redacted] [redacted] 28 OK
To Do	

Specification	**Patni**
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests WriteAssurePolicyDaoTest
Tier	Server
Developer	Lilly [redacted]
Total Number	28
Log	Lilly [redacted] [redacted] 28 OK
To Do	

[redacted]

[redacted]

Specification	**Patni**			
Junit Test Class	Com.ge.gefre.ltca.dao.tests WriteAssureIncrementalTest			
Call Arguments				
Tier	Server			
Developer	Lilly [redacted]			
Total Number	17			
Log	Lilly [redacted]	[redacted]		17 OK
To Do				

Specification	**Patni**			
Junit Test Class	Com.ge.gefre.ltca.dao.tests WriteAsureHistoryPolicyDaoTest			
Call Arguments				
Tier	Server			
Developer	Lilly [redacted]			
Total Number	16			
Log	Lilly [redacted]	[redacted]		16 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests XchangeRate			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	14			
Log	Lilly [redacted]	[redacted]		14 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests TestDelete			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	1			
Log	Lilly [redacted]	[redacted]		1 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests RipRiosResultDAOTest			
Tier	Server			
Developer	Vinod [redacted]			
Total Number	2			
Log	Lilly [redacted]	[redacted]		All OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.dao.tests TestUserInputDAORipRios			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	1			
Log	Lilly [redacted]	[redacted]		1 OK
To Do				

[redacted]

[redacted]

Specification	**Patni**			
Junit Test Class	Com.ge.gefre.ltca.model.tests MovementControllerTest			
Call Arguments				
Tier	Server			
Developer	Vinod [redacted]			
Total Number	4			
Log	Lilly [redacted]	[redacted]		All OK
To Do				

Specification	**Patni**			
Junit Test Class	Com.ge.gefre.ltca.model.tests TestLTCABigDecimal			
Call Arguments				
Tier	Server			
Developer	Lilly [redacted]			
Total Number	11			
Log	Lilly [redacted]	[redacted]		11 OK
To Do				

[redacted]

[redacted]

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.utils.tests TestCalendarHelper			
Tier	Server			
Developer	Lilly [redacted]			
Total Number	12			
Log	Lilly [redacted]	[redacted]		12 OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.tests AllTests			
Tier	Server			
Developer	Vinod [redacted]			
Total Number	This will execute all the Accounting engine test programs.			
Log	Lilly [redacted]	[redacted]		All OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	Com.ge.gefre.ltca.ae.tests AllTestsRipRios			
Tier	Server			
Developer	Vinod [redacted]			
Total Number	This will execute all the test programs, related to RipRios Calculation.			
Log	Lilly [redacted]	[redacted]		All OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	com.ge.gefre.ltca.dao.tests AllTestsRipRiosDAO			
Tier	Server			
Developer	Vinod [redacted]			
Total Number				
Log	Vinod [redacted]	[redacted]		OK
To Do				

[redacted]

[redacted]

Specification	**Patni**			
Junit Test Class Call Arguments	com.ge.gefre.ltca.model.tests AllTestsRipRiosModel			
Tier	Server			
Developer	Vinod [redacted]			
Total Number				
Log	Vinod [redacted]	[redacted]		OK
To Do				

Specification	**Patni**			
Junit Test Class Call Arguments	com.ge.gefre.ltca.ae.tests AllTestsRipRios			
Tier	Server			
Developer	Vinod [redacted]			
Total Number				
Log	Vinod [redacted]	[redacted]		OK
To Do				

[redacted]

[redacted]

8 Standard Operation Procedures (SOP)

8.1 Load & Refresh Database

Step	Description
1	Switch to P:\Shared Folders\LTC Accounting\3_Design\db\Softclose\OracleLoad\Rvrepe
2	Zip the old version RVREPE in the sub folder Load as backup.
3	Ask Kurt[redacted] for the newest Version of the RVREPe. Copy this file in the Load folder.
4	Confirm with the user community the delete & update of the tables RVREPE, CONTRACT_DETAIL, CONTRACT_INDEX, REINSTATEMENT_CONDITIONS, PREMIUM_SHARES,CLAIM, CLAIMS_BOOKING_PAID, CLAIMS_BOOKING_OS, RIP_RIOS_RESULT, RIP_RIOS_RESULT_CURRENCY_SPLIT,AESESSION,AEBOOKING,
5	Execute RvrepeDropCreateGrant.sql
6	Execute LoadRVREPE.bat (approx. 1 hour) Known issues: The NLS variables could cause problems with the decimal separators in numbers. Furthermore, did we receive changing formats of RVREPE (e.g. Fixed Table column width)
7	Execute a SELECT COUNT(*) FROM RVREPE The outcome should be greater than 934529 rows
8	Execute the Index SQLs: (add it to the vss.) CreateIndexRVREPE_CONTRACT_PRIM.sql CreateIndexRvrepeKENZ.sql CreateIndexRVREPE_CATCODE.sql CreateIndexRVREPE_CONTREF.sql CreateIndexRVREPEClaimno.sql CreateIndexContractPrimKey.sql CreateAllIndex.sql (All Cognos Cube related Indicies)
9	A 1:1 table represent all bookings of RVREPE in USD. Therefore we have to execute the SQL "2608Dycreate rvrepe_amountusd.sql" and SQL 2608Dyinsert_into_rvrepe_amountusd.sql

[redacted]

10	Check the state of the CURRENCY table: Do this table contains the Currencies of the current month? SELECT * FROM CURRENCY WHERE MONTH = 8 AND YEAR = 102
11	Execute a SELECT COUNT(*) FROM BUSINESS_ADDL_WA The outcome should be greater than 211602 rows
12	Confirm with the user community all booking codes of 'Claims Paid' & 'Claims Outstanding'. If the list of KENZ is changed, please manipulate the following SQLs accordingly. See Table ACCOUNT_MAPPING
13	Create CLAIMS_BOOKING_PAID and CLAIMS_BOOKING_OS (see SQL CreateClaimBookingOS.sql and CreateClaimBookingPaid.sql)
14	Execute the LoadHistoryTable.java to extract all the records out of the RVREPE and the BUSINESS_ADDL_WA tables.All the input tables relevant to the Rip/Rios calculator will be loaded/updated with the latest dump of the RVREPE and the BUSINESS_ADDL_WA tables.The tables which would be populated are LOAD_HISTORY, CONTRACT_DETAIL,CONTRACT_INDEX,PREMIUM_SHARES,REINSTATEMENT_CONDITIONS,CLAIM,CLAIMS_BOOKING_PAID and CLAIMS_BOOKING_OS . BUSINESS_ADDL_WA would load the CONTRACT_DETAIL and the REINSTATEMENT_CONDITIONS tables. The RVREPE would load CONTRACT_INDEX, CLAIM, CLAIMS_BOOKING_PAID and CLAIMS_BOOKING_OS LOAD_HISTORY is feeded when the LoadHistoryTable.java program is invoked.
15	Check the state of the CONTRACT_DETAIL table. Are their any additional Contracts in Writasure? Are any of contract parameter changed? Load the newest version if any delta is obvious.The load program is to be run again.
16	Inform Cognos Support and demand a Cube refresh.
17	Execute a SELECT COUNT(*) FROM CONTRACT_DETAIL The outcome should be greater than 41695 rows Execute the following SQL and make sure that you only get one row: SELECT * FROM CONTRACT_DETAIL WHERE MAJORCLASS_NAME = 'N' AND CONTRACT_NBR = 449 AND UWYR_YR = 2000 AND VERSION_NAME = '0' AND ENDORSE_NAME = '00'
18	Execute a SELECT COUNT(*) FROM CONTRACT_INDEX

	The outcome should be greater than 5282 rows Execute the following SQL and make sure that you only get one row: SELECT * FROM CONTRACT_INDEX WHERE FK_MAJORCLASS = 'N' AND FK_CONTRACT = 449 AND FK_UWYR = 2000 AND FK_VERSION = '0' AND FK_ENDORSE = '00'
19	Execute the following SELECT COUNT(*) FROM REINSTATEMENT_CONDITIONS WHERE REINST_ID = 1 AND FK_LOAD_ID = 1 The outcome should be greater than 15534 rows SELECT COUNT(*) FROM REINSTATEMENT_CONDITIONS WHERE REINST_ID = 2 AND FK_LOAD_ID = 1 The outcome should be greater than 740 rows SELECT COUNT(*) FROM REINSTATEMENT_CONDITIONS WHERE REINST_ID = 3 AND FK_LOAD_ID = 1 The outcome should be greater than 139 rows SELECT COUNT(*) FROM REINSTATEMENT_CONDITIONS WHERE REINST_ID = 4 AND FK_LOAD_ID = 1 The outcome should be greater than 44 rows
20	Execute a SELECT COUNT(*) FROM CLAIM The outcome should be greater than 24419 rows Execute the following SQL and make sure that you only get one row: SELECT * FROM CLAIM WHERE FK_MAJORCLASS = 'N' AND FK_CONTRACT = 449 AND FK_UWYR = 2000 AND FK_VERSION = '0' AND FK_ENDORSE = '00'
21	Execute a SELECT COUNT(*) FROM CLAIMS_BOOKING_PAID

[redacted] [redacted]

	The outcome should be greater than 92574 rows
22	Execute a SELECT COUNT(*) FROM CLAIMS_BOOKING_OS The outcome should be greater than 115832 rows
23	Execute the following SQL SELECT SUM(REINST_NBR) FROM REINSTATEMENT_CONDITIONS WHERE FK_CONTREF_ID = 'C-000417-1992-0-00' AND FK_LOAD_ID = 1 Execute the following SQL and make sure that the **reinst_cnt** is equal to the **total sum of the reinst_nbr** for that particular contract. SELECT REINSTS_CNT FROM CONTRACT_DETAIL WHERE CONTREF_ID = 'C-000417-1992-0-00' AND AND FK_LOAD_ID = 1 The result of both these queries should be the same for a particular load id.
24	Execute a SELECT COUNT(*) FROM CONTRACT_INDEX WHERE MAJORCLASS_NAME = 'C' The outcome should be greater than 423 rows Execute a SELECT COUNT(*) FROM CONTRACT_INDEX WHERE MAJORCLASS_NAME = 'M' The outcome should be greater than 720 rows Execute a SELECT COUNT(*) FROM CONTRACT_INDEX WHERE MAJORCLASS_NAME = 'N' The outcome should be greater than 3726 rows
25	SELECT COUNT(*) FROM CONTRACT_DETAIL WHERE EXPIRYDATE_DATE IS NULL The outcome is 9 rows SELECT COUNT(*) FROM CONTRACT_DETAIL WHERE INCEPTIONDATE_DATE IS NULL The outcome is 9 rows

26	Execute a SELECT COUNT(*) FROM CLAIMS_BOOKING_OS WHERE KENZ = 4351 The outcome should be greater than 115832 rows Execute a SELECT COUNT(*) FROM CLAIMS_BOOKING_OS WHERE KENZ = 4204 The outcome should be 0 Rows
27	Execute a SELECT COUNT(*) FROM CLAIMS_BOOKING_PAID WHERE KENZ = 4204 The outcome should be greater than 92114 rows Execute a SELECT COUNT(*) FROM CLAIMS_BOOKING_PAID WHERE KENZ = 4351 The outcome should be 0 Rows
28	The config folder should be present in the current directory, and it should contain the RipRiosRunLogConfig.txt file. This config file must define a RollingFileAppender for the root category. This is accessed while initializing the log environment for the application run.

8.2 Run RIP/RIOS Accounting Engine

Requirements

JUnit 3.7

Oracle JDBC driver classes12.zip

Call of RIP/RIOS Accounting Engine

java com.ge.gefre.ltca.AppMain –riprios -user <user-name> -cutoffdate <dd/mm/yyyy> -exchangeratedate <dd/mm/yyyy>	
OR java com.ge.gefre.ltca.AppMain –ripriosHistory -user <user-name> -StartDate <dd/mm/yyyy> EndDate <dd/mm/yyyy> -exchangeratedate <dd/mm/yyyy>	
-user	The name will be used to create a Session key. Please insert your unique name. You could also concatanete some explaining text in order to describe the particular run e.g. `Dan-Q4-1998, Job-NewDataLoad-Q3-2002` **(Mandatory)**
-cutoffdate	All bookings with a billing month and billing year less equal than the given date will be selected. Currently, the day isn't important in the Revenue Report scenario, but in a RDB environment. **(Mandatory for -RipRios)**
- exchangeratedate	We use the default Q2/2002 as Booking level currency exchange dates. If you want to specify any other date for testing purpose, use this flag. **(Non Mandatory)** Default: Q2/2002
-exchangetable	This specifies the exchangerate table to be used for getting the currency exchange rates for the calculations. **(Non Mandatory)** Default: ge_exchange_rate.
-policy	This specifies the policy to be used in case the exchangerate is not found. **(Non Mandatory)** Default: Exception policy.
-sessiontype	This parameter specifies the type of run. (Example productive or temp). **(Non Mandatory)** Default: None
-usercomment	Through this parameter the user can feed his comments before commencing the run. **(Non Mandatory)** Default: None

[redacted] [redacted]

-loadid	Through this field the user can specify the specific load of the database that the calculator should use for the calculations. **(Non Mandatory)** Default: The most recent load is used as the default.
-previoussession	This parameter allows the user to specify the session, which should be used for movement calculations.**(Non Mandatory)** If the user specified session is invalid the calculator is halted. If this is not specified Balance is inserted as Movement.
- contractratepolicy	Can specify true or false. If this value is false, user specified / default exchange rate date will be used instead of Inception date, for Contract level conversions. **(Non Mandatory)**
–whereclause	Where clause for Contract fetch. The Contract fetch can be limited by this where clause. **(Non Mandatory)**
-startdate	This is a mandatory parameter for the RipRiosHistory run. It specifies the date from which cutOffDate generation starts.**(Mandatory for RipRiosHistory)**
-enddate	This is a mandatory parameter for the RipRiosHistory run. It specifies the date upto, which the cutoff dates, are to be generated. **(Mandatory for RipRiosHistory)**
-noauditsummary	This is a flag that specifies whether to generate RipRiosResult.xml. If it is specified then ripriosresult.xml is not generated. **(Non Mandatory)**
-noauditcontract	This is a flag that specifies whether to generate Contract xmls. If it is specified then contract XMLs are not generated. **(Non Mandatory)**
-whereclause	This is used to restrict the number of Contracts for RipRios calculation. The WhereClause has to be in proper SQL format. **(Non Mandatory)**

Example:

```
java com.ge.gefre.ltca.AppMain -RipRios
-User -Job-V1.05-Q4-2001 -CutOffDate 30/12/2001
-ExchangeRateDate 30/12/2001 -ExchangeTable currency
-Policy ExceptionPolicy -SessionType productive
-UserComment "Quarter 2 Run" -LoadId 1 –PreviousSession 1073
-ContractratePolicy [true/false] -WhereClause "CI.UWYR_YR = 1976"
-StartDate <dd/mm/yyyy> -EndDate <dd/mm/yyyy>
```

For convenience, you can use the following batch:

RunDb.bat %1 %2

java com.ge.gefre.ltca.AppMain -RipRios -User %1 -CutOffDate %2

Exchange Rate Tables Available:

Table Name	Description
Currency	Exchange rates are retrieved from the WritAsure Exchange Rate Table
ge_exchange_rate	Exchange rates are retrieved from the [redacted] Exchange Rate Table
Sap_rates	Exchange rates are retrieved from the sap exchange Rate Table

Exchange Rate Policies Available:

Policy Name	Description
Exception Policy	If Exchange Rate for the specified date does not exist, Exception is raised.
History	If Exchange Rate for the specified date does not exist, search is

[redacted]

[redacted]

Policy	done by going back in history through all records, till it finds a rate. If rate is still not found, Exception is thrown.
Incremental Policy	If Exchange Rate for the specified date does not exist, search is done on Daily, Monthly, Quarterly and Yearly basis. If rate is still not found, Exception is thrown
Sap Policy	This has to be used with Sap_Rates table. If Exchange Rate Date specified is Q4 2001 or less, Q4 2001 rates will be used, else rates for specified date will be used. If rates are not found for these dates, Exception is thrown.

8.3 Build RIP/RIOS Accounting Engine

Switch to the Build folder of the current version and delete the folder build**build** here. Make sure, that all settings in BUILD.BAT are valid for your box. Run the following:

```
build core -Dversion=1.03
```

[redacted]

[redacted]

2 Core Data Tables

The source, destination and intermediated tables needed by the algorithm are described here. All external tables referenced by the IBNR Generator are described by the LTC_Report Scheme and are found in DLTC-E.

2.1 Treaty Header

Treaty header data is within the table LTC_Report.Treaty_Header. For completeness the entire table is shown below with fields required by the IBNR Generator marked with an "x".

Treaty Header as of 23 Apr 2003			
Field	Format	Required	
EST_DIM_I	number		
DELETED	umber(1,0)	x	
MODIFICATION-DATE	date		
ACTION	varchar2(32)		
FK_SESSION	integer		
BUS_DIM_I	number(10,0)		
SRC_SYS_N	varchar2(9)	to book	
ORIG-BUS-REIN-C	varchar2(5)	to book	
CDNT_I	varchar2(7)	to book	
BRKR_I	varchar2(7)	to book	
DIR_INDR_F	varchar2(1)	to book	
UWG_YR_D	number(5,0)	to book	
METH_PLACING_C	varchar2(3)		
INCEPTION_D	date	x	
EXPIRY_D	date	x	
CUR_C	varchar2(5)	x	
SEC_NUM_Q	number(3,0)		
MAXLIMIT	number(13,0)		
CNTRCT_STUS_C			
CNTRCT_SYS_STUS_C			
TRTY_NUM_I	varchar2(12)	x	
ESTIMATED LOSS			
CUWY_STATUS			
RISKTYPE_C	varchar2(5)		
MAX_ACTG_YR_D			
ORIG_EPI_A			
ORIG_IBNR_A			
ORIG_COM_A			
ORIG_EP_A			
ORIG_RESULT_A			
ACTG_BSIS_C	varchar2(15)	x	Cover Basis stored here. FORMAT?
CLM_BSIS_C			
BUS_PRTF_I	varchar2(5)	x	Is this the portfolio number or index???

[redacted]

2.2 *Claims Header*

Claims header data is within the table LTC_Report.CLAIMS.. For completeness the entire table is shown below. Those fields required by the IBNR Generator Algorithm are marked with an "x".

CREATE TABLE Claim			
25-Apr-03			
ID	NUMBER(*,0		
CLAIMNO	VARCHAR2(18))	x	ask HJ for needed values
DATEofLOSS	DATE	x	
YEARofLOSS	VARCHAR2(4)	x	
CATCODE	VARCHAR2(6)	x	
PERIL	VARCHAR2(3)		
~~FK_BUS_DIM_I~~	~~VARCHAR2~~		deleted
FK_MAJORCLASS	varchar2(1)		
FK_CONTRACT	number(6,0)		
FK_UWYR	number(4,0)		
FK_VERSION	varchar2(2)		
FK_ENDORSE	varchar2(2)		
FK_CONTREF	VARCHAR2(18))		

2.3 *LTC_Ledger*

LTC report contains a ledger similar to RDB's accounting details, except

- The original booking codes are replaced with the new ACE Booking code groups (same level of detail as BP_Label).
- The table does not contain as many fields as the original Accounting details.

The table description follows.

Table LTC_REPORT.LEDGER 8-Apr-2003 21:32:52 from LTC_REPORT@dice.WORLD	
LEDGER_DIM_I	number
BUS_DIM_I	number(10,0)
SEC_NUM_Q	number(3,0)
TRTY_NUM_I	varchar2(12)
UWG_YR_D	number(5,0)
LOSSYEAR	number(4,0)
CLAIM_NUMBER	number(7,0)
BOOKING_CODE_C	varchar2(5)
CUR_C	varchar2(5)
AMOUNT_A	number(18,2)
GBL_A	number(18,2)
SOURCE_C	varchar2(32
INWD_OWRD_C	varchar2(7)
FY	number(4,0)
FY_MONTH	number(2,0)
FY_QUARTER	varchar2(2)
GERMAN_GAAP_REV_YEAR	number(4,0)
IBNR	number
ESTIMATED_LOSS	number
CMPR_D	date
~~Booking Date~~	Requested by HJ
~~Billing Year~~	~~Requested by HJ~~
~~Billing Month~~	~~Requested by HJ~~

will be deleted?

RDB- booking date..still missing

Note:, booking_Year, booking_Month are still missing
Do we need claim table key?
What does Q at the end of a field name mean?

To get to Treaty_Header I use: BUS_DIM_I + SEC_NUM_Q + end_of_period_date

[redacted]

2.4 AE_Bookings

All output of the Loss Year splitter goes to AE_Bookings (which has the same format as the RDB accounting details.)

The Loss year splitter will generate "bookings" which are stored in the Treaty booking ledger table (which later goes to SAP) as well as the input table.

Field Name	Source As discussed with Client
Id	Sequence in Oracle Uniquely identifies a record in the output table.
SessionId	FK from Session Table
Source	[SOURCE_SYS_N] LAST_ULT_STUS.SRC_SYS_N or
reinsure	[Reinsurer] LAST_ULT_STUS.ORIG_BUS_REIR_C
book_id	See Details :
Uw_year	[uw_year] BUSINESS. B.UWG_YR_D
Occ_year	0
Account_year	[current Year]
Account_period	Y1
BYRP	[curentyear]
book_Branch	Blank
Bookcode	[bookcode] O/p of the IBNR
orig_curr	[Main Currency] LAST_ULT_STUS.BUS_CUR_C
Amount	[Amount] O/p of the IBNR

[redacted]

	If the amount calculated has a negative sign , put the signed amount in this field and keep the Sign field blank.
Sign	[sign] blank
A_type	E
claim_id	Blank
Pool_id	I Blank
Refno	Blank
Ref_type	TECHNICAL ACCNT
computer_date	[redacted]
Posting_date	[redacted]
treatyno	[treatyno] Business.TRTY_NUM_I
in_out	INWARD
Dir_indir	I BUSINESS_ATTR_1.Dir_indir
branch	[branch] Lasr_Ult_Stus.Trty_Bus_Cls_C
pool_re	Blank
Cedent	[cedent] Business.cedent
Broker	[broker] Business.broker
Sap_comp	Blank
Bus_area	Blank
Agg_code	Blank
SAP_type	Blank
SAP_branch	Blank
trading_part	Blank
Bank_account	Blank
SAP_cur	Blank

[redacted]

Sap_aggr	YES
orig_computer_date	[redacted]
Lirma_ref	Blank
lirma_f	N
[redacted]	[redacted]

2.5 *AE_Session*

The Accounting engine, like all others, must store session information here. This information will be used only for roll back.

[redacted]	[redacted]
Id	Primary Key Identifying a Session
DateOfRun	Current Date
TimeTag	It gives the Time Stamp
SessionType	
SoftwareVersion	This gives the version of the software.
UserName	Input to the calculator
UserComment	Input to the calculator

2.6 Portfolio Loss Ratio History Tables

The IBNR generator requires as input an expected loss ratio for each portfolio per loss year. These ratios could have two sources:

- Session 2 planning
- Reserve Pro

Regardless of the source the data will be entered by hand via the Loss Ratio Entry module described in a separate document. The Loss Ratio Entry module owns the tables:

Loss Ratio History View			
Col no	**Name**	**Format**	**Comments**
1	Portfolio_ID (primary key)	integer	Link for Port. Names, Descp.
2	SeqNum (primary key)	integer	Autoincremented sequential number
3	Loss_Year	Number (4)	
4	Loss_Ratio	Decimal(10, 2)	
5	Entry_Date	Date	
6	Booking_Comment	Text (20)	Booker's General comment indicating reason
7	SSO_ID	Integer	Link for SSO, names, initials

Note: the names and groupings of the portfolios are identical to those used by other engines. The loss Ratios are "balances".

Portfolio Description Table			
Col no	**Name**	**Format**	**Comments**
1	Portfolio_ID (key set)	integer	
2	Portfolio_Name	Text (6)	
3	Portfolio_Description	Text (30)	

From these two tables the IBNR generator needs a view which gives the "as of" values sorted by

- Portfolio name
- Loss Year
- Loss Ratio

[redacted]